Package 'RAM'

August 21, 2014

Type Package

Title R for Amplicon-based Metagenomics

Version 1.0.0

Date 2014-08-20

Author Wen Chen and Joshua Simpson

Copyright Government of Canada

Maintainer Wen Chen <Wen.Chen@agr.gc.ca>

Description This package provides a series of functions to make amplicon-based metagenomic analysis more accessible, and publication-quality plots simple. Amplicon-based (or targeted) metagenomics amplifies and sequences selected DNA regions of environmental samples, but not the entire pool of genetic material, which is referred to as shotgun metagenomics. The amplicon-metagenomics mainly aims at characterizing broad microbiota biodiversity in different environments.

License MIT + file LICENSE

Depends vegan, ggplot2, stats

Imports gridExtra, RColorBrewer, indicspecies, Heatplus,splitstackshape, gplots, plyr, reshape2, scales, labdsv, grid,stringr, ggmap, permute

Suggests testthat

R topics documented:

`RAM-package` *Analysis of Amplicon-Based Metagenomic Data*

Description

The RAM package provides a series of functions to make amplicon-based metagenomic analysis more accessible. The package is designed especially for those who have little or no experience with R. This package calls heavily upon other packages (such as `vegan` and `ggplot2`), but the functions in this package either extend their functionality, or increase the ease-of-use.

Details

Package:	RAM
Type:	Package
Version:	1.0.0
Date:	2014-08-20
License:	MIT License, Copyright (c) 2014 Government of Canada

Load data from .csv-formatted OTU files with `read.OTU`, then process the data with other commands. Type the command `library(help = RAM)` for a full index of all help topics, or `ls("package:RAM")` to get a list of all functions in the package.

Type `data(ITS1, ITS2, meta)` to load a sample (fungal) data set.

Type `citation("RAM")` for how to cite this package.

This pacakge contains information licensed under the Open Government Licence - Canada. See `group.spatial` for further details.

Author(s)

Wen Chen and Joshua Simpson.

Maintainer: Wen Chen <wen.chen@agr.gc.ca>

See Also

`vegan`, `ggplot2`

Examples

```
## Not run:
# load data from your own files...
ITS1 <- read.OTU("path/to/OTU/table")
ITS2 <- read.OTU("path/to/OTU/table")
meta <- read.meta("path/to/meta/table")
## End(Not run)

# ...or use the included sample data
data(ITS1, ITS2, meta)

dissim.heatmap(ITS1, meta, row.factor=c(City="City"))
dissim.alleig.plot(ITS1, ITS2)

# type library(help = RAM) to get a full listing of help documents
```

`assist.ordination` *Perform CCA and RDA Analysis for OTU Tables*

Description

This function simplifies CCA and RDA analysis by abstracting away some of the complexity and returning a list of useful measures.

Usage

```
assist.cca(otu1, otu2 = NULL, meta, full = TRUE, exclude = NULL, rank)
assist.rda(otu1, otu2 = NULL, meta, full = TRUE, exclude = NULL, rank)
```

Arguments

otu1	the first OTU table to be used.
otu2	the second OTU table to be used.
meta	the metadata table to be used (must have same samples as otu1/otu2).
full	logical. Should a full model be considered? (If not, a restricted model is used).
exclude	A vector, either numeric or logical, specifying the columns to be removed from meta. If a character vector, columns with those names will be removed; if a numeric vector, columns with those indices will be removed.
rank	a character vector representing a rank. Must be in one of three specific formats (see ?RAM.rank.formatting for help).

Value

If both otu1 and otu2 are given, a list of length 2 will be returned with the following items (if only otu1 is given, a list of length 1 will be returned with these items):

$GOF	the goodness of fit scores for the model.
$VIF	the VIF scores for the model.
$percent_variation	the percent variation explained by each axis
$CCA_eig	Eigenvalues for CCA axes.
$CA_eig	Eigenvalues for CA axes.
$anova	the ANOVA results for the model.

Author(s)

Wen Chen and Joshua Simpson.

See Also

cca, anova.cca

Examples

```
data(ITS1, meta)

cca.help <- assist.cca(ITS1, meta=meta, rank="p")
cca.help$anova
```

dissim *Calculate Dissimilarity Matrix Data*

Description

These functions calculate different measures related to dissimilarity matrices. All of these functions allow you to specify one of many dissimilarity indices to be used.

Usage

```
dissim.clust(data, dist.method="morisita", clust.method="average")
dissim.eig(data, method="morisita")
dissim.ord(data, dist.method="morisita", k=NULL)
dissim.GOF(data, method="morisita")
dissim.tree(data, dist.method="morisita", clust.method="average")
dissim.pvar(data, method="morisita")
```

Arguments

data	the OTU table to be used.
method	the dissimilarity index to be used; one of "manhattan", "euclidean", "canberra", "bray", "kulczynski", "jaccard", "gower", "altGower","morisita", "horn", "mountford", "raup", "binomial", "chao", or "cao".
dist.method	same as method, but named differently for functions with other method parameters.
k	the number of dimensions desired. If NULL, the maximum value will be calculated and used.
clust.method	the method used for clustering the data. Must be one of "ward", "single", "complete", "average", "mcquitty", "median", or "centroid".

Value

dissim.clust	returns a hierarchical clustering of the dissimilarity matrix.
dist.eigenval	returns the eigenvalues of the dissimilarity matrix.
dissim.ord	returns a list: the first item is the the ordination distances, the second is the dissimilarity matrix distances.
dissim.GOF	returns the goodness of fit values of the dissimilarity matrix, for various numbers of dimensions used.
dissim.tree	returns a list: the first item is the tree distances, the second is the dissimilarity matrix distances.
dissim.pvar	returns a numeric vector containing the percent variation explained by each axis (where each sample corresponds to an axis).

Author(s)

Wen Chen and Joshua Simpson

See Also

vegdist, hclust, dissim.plot

Examples

```
data(ITS1)

# calculate clustering, using default method
dissim.clust(ITS1)

# calculate tree distances, specifying a distance method
# (but use default clustering method)
dissim.tree(ITS1, dist.method="euclidean")

# calcualte ordination distances, specifying both distance
# and ordination methods
dissim.ord(ITS1, dist.method="bray", k=3)
```

---

dissim.heatmap *Plot Distance Matrix Heatmap for OTU Samples*

Description

This function consumes an OTU table, along with some optional parameters, and creates a heatmap showing the distance matrix for the samples of the given OTU table.

Usage

```
dissim.heatmap(data, meta=NULL, row.factor=NULL, col.factor=NULL,
               stand.method="chi.square", dissim.method="euclidean",
               file=NULL, ext=NULL, height=8, width=9, leg.x=-0.05, leg.y=0)
```

Arguments

data	the OTU table to be used.
meta	the metadata table to be used.
row.factor	a factor from the metadata to show along the rows of the heatmap (see Details below).
col.factor	a factor from the metadata to show along the columns of the heatmap (see Details below).
stand.method	a method used to standardize the OTU table. One of "total", "max", "freq", "normalize", "range", "standardize", "pa", "chi.square", "hellinger" or "log" (see ?decostand).

dissim.method	the dissimilarity index to be used; one of "manhattan", "euclidean", "canberra", "bray", "kulczynski", "jaccard", "gower", "altGower","morisita", "horn", "mountford", "raup", "binomial","chao", or "cao" (see ?vegdist).
file	the file path where the image should be created (see ?RAM.plotting).
ext	the file type to be used; one of "pdf", "png", "tiff", "bmp", "jpg", or "svg".
height	the height of the image to be created (in inches).
width	the width of the image to be created (in inches).
leg.x	how far the legend should be inset, on the x axis.
leg.y	how far the legend should be inset, on the y axis.

Details

Both row.factor and col.factor should be named character vectors specifying the names of the rows to be used from meta (see RAM.factors). They should also be factors; if they are not, a warning is raised and they are coerced to factors (see factor). A warning is also raised when a factor has more than 8 levels, as that is the most colours the current palettes support. The factor must also correspond to the OTU table; i.e. they should have the same samples. If not, an error is raised.

Note

This function creates the heatmap using the heatmap.2 function from the gplots package. That function calls layout to set up the plotting environment, which currently prevents plotting two data sets side by side, or to automatically place the (metadata) legend. Unfortunately, this means that the leg.x and leg.y parameters must be used to adjust the legend by trial and error. It is possible to move the legend outside of the plotting area; if not legend appears, try with small leg.x and leg.y values.

Author(s)

Wen Chen and Joshua Simpson.

See Also

decostand, vegdist, factor, heatmap.2, RAM.factors

Examples

```
data(ITS1, meta)

# plot to the screen with one meta factor and standard calculation methods
dissim.heatmap(ITS1, meta, row.factor=c(Plot="Plots"))

## Not run:
# plot the heatmap to a .tiff file using Hellinger standardization and Manhattan
# distances
dissim.heatmap(ITS1, dissim.method=manhattan, stand.method=hellinger,
                    file="my/sample/path", ext="tiff")
## End(Not run)
```

dissim.plot *Plot Dissimilarity Matrix Data for Different Methods*

Description

These functions all produce a plot of some measure related to dissimilarity matrices. All of these functions allow you to specify a vector of methods to be used when creating the plot.

Usage

```
dissim.clust.plot(otu1, otu2=NULL, file=NULL, dist.methods=NULL,
                  clust.methods=NULL)
dissim.eig.plot(otu1, otu2=NULL, file=NULL, dist.methods=NULL)
dissim.alleig.plot(otu1, otu2=NULL, file=NULL, dist.methods=NULL)
dissim.ord.plot(otu1, otu2=NULL, file=NULL, dist.methods=NULL, k=NULL)
dissim.GOF.plot(otu1, otu2=NULL, file=NULL, dist.methods=NULL)
dissim.tree.plot(otu1, otu2=NULL, file=NULL, dist.methods=NULL,
                 clust.methods=NULL)
dissim.pvar.plot(otu1, otu2=NULL, file=NULL, dist.methods=NULL)
```

Arguments

otu1	the otu1 data to be used.
otu2	the otu2 data to be used.
file	the file path for the plot. If not provided (defaults to NULL), then the plot is displayed to the screen. If file is provided, that is where the .tiff file will be created.
dist.methods	a character vector representing the dissimilarity indices to be used; each element must be one of one of "manhattan", "euclidean", "canberra", "bray", "kulczynski", "jaccard", "gower", "altGower","morisita", "horn", "mountford", "raup", "binomial","chao", or "cao".
clust.methods	a character vector representing the methods used for clustering the data. Each element must be one of "ward", "single", "complete", "average", "mcquitty", "median", or "centroid".
k	the number of dimensions desired. If NULL, the maximum value will be calculated and used.

Details

All of these functions (other than dissim.alleig.plot) call their dissim.X counterparts and plot the data. If file is given, a .tiff file will be created at file; otherwise the plot is displayed to the screen.

Value

All functions create a plot and return the plotted data invisibly.

`dissim.clust.plot`
: plots a hierarchical clustering of the dissimilarity matrix.

`dissim.eig.plot`
: plots a bar plot of the eigenvalues of the dissimilarity matrix.

`dissim.alleig.plot`
: plots a line plot showing the relative importance of all eigenvalues for a variety of methods.

`dissim.ord.plot`
: plots a scatter plot comparing the "euclidean" distances among all samples in ordination space to the dissimilarity matrix distances.

`dissim.GOF.plot`
: plots a scatter plot of the goodness of fit values of the dissimilarity matrix, for various numbers of dimensions used.

`dissim.tree.plot`
: plots a scatter plot comparing the tree distances to the dissimilarity matrix distances.

`dissim.pvar.plot`
: plots a bar plot showing the percent variation explained by each axis (where each sample corresponds to an axis).

Note

If `file` does not end in ".tiff", then ".tiff" will be appended to the end of `file`.

Function `dissim.alleig.plot` uses the `ggplot2` package for creating the plot, and returns the plot object. This means that you can store this plot and add other features manually, if desired (see Examples).

Author(s)

Wen Chen and Joshua Simpson

See Also

`vegdist`, `hclust`, `dissim`, `ggplot`

Examples

```
data(ITS1, ITS2)

# show percent variation for only ITS1 with default methods
dissim.pvar.plot(ITS1)

# show clustering for ITS1 and ITS2 for set methods
dissim.clust.plot(ITS1, ITS2, dist.methods=c("morisita", "bray"),
                  clust.methods=c("average", "centroid"))
```

```
## Not run:
# dissim.alleig.plot returns a ggplot2 object:
my.eig.plot <- dissim.alleig.plot(ITS1, ITS2)
class(my.eig.plot) # returns "gg" "ggplot"
my.eig.plot # view the plot
# update the title, then view the updated plot
my.eig.plot <- my.eig.plot + ggtitle("My New Title")
my.eig.plot

# save an image (named file.tiff) with GOF values for ITS1 and ITS2, using
# default methods
dissim.GOF.plot(ITS1, ITS2, file="~/Documents/my/file")
## End(Not run)
```

`diversity.indices` *Calculate True Diversity and Evenness*

Description

These functions calculate true diversity and evenness for all samples.

Usage

```
true.diversity(otu1, otu2 = NULL, index = "simpson")
evenness(otu1, otu2 = NULL, index = "simpson")
```

Arguments

`otu1`	the otu1 data to be processed.
`otu2`	the otu2 data to be processed.
`index`	the index to use for calculations; partial match to "`simpson`" or "`shannon`".

Details

The `otu2` argument is optional; if it is not specified, the calculations will be done for only the `otu1` argument.

For the following sections, S represents the number of species, λ represents the Simpson index, and H' represents the Shannon index.

The formulas for the true diversity of the indices are as follows:

- Simpson: $D_2 = frac1\lambda$
- Shannon: $D_1 = \exp H'$

The formulas for the evenness of the indices are as follows:

- Simpson: $\frac{\frac{1}{\lambda}}{S}$
- Shannon: $\frac{H'}{\ln S}$

Value

Both functions return a numeric matrix, where the rows are the given OTUs, and the columns are the samples.

Note

Credit goes to package vegan for the partial argument matching (see References).

Author(s)

Wen Chen and Joshua Simpson.

References

Jari Oksanen, F. Guillaume Blanchet, Roeland Kindt, Pierre Legendre, Peter R. Minchin, R. B. O'Hara, Gavin L. Simpson, Peter Solymos, M. Henry H. Stevens and Helene Wagner (2013). vegan: Community Ecology Package. R package version 2.0-10. http://CRAN.R-project.org/package=vegan

Diversity index. (2014, May 7). In Wikipedia, The Free Encyclopedia. Retrieved 14:57, May 28, 2014, from http://en.wikipedia.org/w/index.php?title=Diversity_index&oldid=607510424

Blackwood, C. B., Hudleston, D., Zak, D. R., & Buyer, J. S. (2007). Interpreting ecological diversity indices applied to terminal restriction fragment length polymorphism data: insights from simulated microbial communities. Applied and Environmental Microbiology, 73(16), 5276-5283.

Examples

```
data(ITS1, ITS2)

# true diversity, using default index (Simpson)
true.diversity(ITS1)

# true diversity for ITS1 and ITS2, using Shannon
true.diversity(ITS1, ITS2, index="shannon")

# default evenness (Simpson) for ITS1/ITS2
evenness(ITS1, ITS2)

# Shannon evenness
evenness(ITS1, index="shannon")
```

---

get.rank *Get OTUs Classified at Taxonomic Rank(s)*

Description

This function returns the OTUs of the given OTU table(s) which are classified at the given taxonomic rank.

Usage

```
get.rank(otu1, otu2 = NULL, rank = NULL)
```

Arguments

otu1	the first OTU table to be used.
otu2	the second OTU table to be used.
rank	a character vector representing a rank. Must be in one of three specific formats (see RAM.rank.formatting for help). If omitted, the function will repeat for all seven major taxonomic ranks.

Value

The value returned by this function may become nested lists, so for convenience, any nested lists have been given descriptive items names to make accessing its elements simple (see Examples).

- If otu2 is given:
  - If rank is given: a list containing two data frames (otu1 and otu2 selected at the given rank).
  - If rank is not given: a list containing two lists. The first sublist represents otu1, the second otu2. The sublists contain seven data frames, which are the OTU tables selected at each taxonomic rank (see Examples).
- If otu2 is not given:
  - If rank is given: a single data frame (otu1 selected at the given rank).
  - If rank is not given: a list containing seven data frames (otu1 selected at each taxonomic rank).

Author(s)

Wen Chen and Joshua Simpson.

Examples

```
data(ITS1, ITS2)

# the following are equivalent:
ITS1.p <- get.rank(ITS1, rank="p")
# this list has get.rank(ITS1, rank="k"), get.rank(ITS1, rank="p"), ...
lst <- get.rank(ITS1)
stopifnot(identical(ITS1.p, lst$phylum)) # true

# get a list of length 2: the item holds all ITS1 data, the second holds ITS2
# data
lst.all <- get.rank(ITS1, ITS2)
stopifnot(identical(ITS1.p, lst.all$otu1$phylum))
```

`group.abundance` *Plot the Abundance of OTUs by Classification at a Given Taxonomic Rank*

Description

This function consumes an OTU, and a rank, as well as various optional parameters. It creates a stacked bar plot showing the abundance of all classifications at the given taxonomic rank for each sample.

Usage

```
group.abundance(otu1, otu2=NULL, rank,
                top=NULL, count=FALSE, drop.unclassified=FALSE,
                file=NULL, ext=NULL, labels=c("ITS1", "ITS2"),
                height=8, width=16, bw=FALSE, ggplot2=TRUE)
```

Arguments

`otu1`	the first OTU table to be used.
`otu2`	the second OTU table to be used.
`rank`	the taxonomic rank to use (see ?RAM.rank.formatting for formatting details).
`top`	the number of groups to select, starting with the most abundant. If `NULL`, all are selected.
`count`	logical. If `TRUE`, the numerical counts for each OTU will be shown; otherwise the relative abundance will be shown.
`drop.unclassified`	logical. Should unclassified samples be excluded from the data?
`file`	the file path where the image should be created (see ?RAM.plotting).
`ext`	the file type to be used; one of `"pdf"`, `"png"`, `"tiff"`, `"bmp"`, `"jpg"`, or `"svg"`.
`labels`	a character vector giving the labels for the panels of the plot.
`height`	the height of the image to be created (in inches).
`width`	the width of the image to be created (in inches).
`bw`	logical. Should the image be created in black and white?
`ggplot2`	logical. Should the `ggplot2` package be used to produce the plot, or should the base graphics be used? (see ?RAM.plotting).

Author(s)

Wen Chen and Joshua Simpson

Examples

```
data(ITS1, ITS2)

# plot the relative abundance at the class level to the screen, ignoring the
# unclassified
group.abundance(ITS1, rank="class", drop.unclassified=TRUE)

## Not run:
# plot the count abundance at the phylum level to path.tiff
group.abundance(ITS1, ITS2, rank="p", file="my/file/path", ext="tiff",
count=TRUE)
## End(Not run)
```

`group.heatmap` *Plot OTU Abundance at a Given Rank with Metadata Annotation*

Description

This function plots the abundance for all taxon groups at a given rank. Additionally, it can display metadata for the samples as annotations along the rows of the heatmap.

Usage

```
group.heatmap(data, meta, rank, factors,
              top=NULL, count=FALSE, drop.unclassified=FALSE,
              cut=NULL, file=NULL, ext=NULL, width=9, height=9)
```

Arguments

`data`	the OTU table to be used.
`meta`	the metadata table to be used.
`rank`	the taxonomic rank to use (see `RAM.rank.formatting` for formatting details).
`factors`	a named character vector indicating the columns of the metadata table to be used (see `RAM.factors`).
`top`	the number of groups to select, starting with the most abundant. If `NULL`, all are selected.
`count`	logical. If `TRUE`, the numerical counts for each group will be shown; otherwise the relative abundance will be shown.
`drop.unclassified`	logical. Should OTUs labelled "unclassified" or missing classification at the given taxonomic rank be excluded?
`cut`	the height at which to cut the sample tree, this will create distinct coloured groups. Currently this will allow for at most nine groups (see Details).
`file`	the file path where the image should be created (see ?RAM.plotting).
`ext`	the file type to be used; one of `"pdf"`, `"png"`, `"tiff"`, `"bmp"`, `"jpg"`, or `"svg"`.
`height`	the height of the image to be created (in inches).
`width`	the width of the image to be created (in inches).

Details

A warning from `brewer.pal` indicating "n too large, allowed maximum for palette Pastel1 is 9" means that the cut height is too low to allow for that many groups. This should be fixed in a future release.

Author(s)

Wen Chen and Joshua Simpson.

See Also

`decostand`

Examples

```
data(ITS1, meta)

group.heatmap(ITS1, meta, rank="c", factors=c(City="City", Plot="Plots"))
```

---

group.heatmap.simple *Plot a Heatmap Showing OTU Abundance by Taxonomic Classification*

Description

This function consumes an OTU table and a rank, as well as some optional parameters, and creates a heatmap showing the abundance of the OTUs at the given taxonomic rank for each sample.

Usage

```
group.heatmap.simple(data, meta=NULL, rank, row.factor=NULL,
                     top=NULL, count=FALSE, drop.unclassified=FALSE,
                     dendro="none", file=NULL, ext=NULL,
                     width=9, height=8, leg.x=-0.08, leg.y=0)
```

Arguments

`data`	the OTU table to be used.
`meta`	the metadata table to be used.
`rank`	the taxonomic rank to use (see ?RAM.rank.formatting for formatting details).
`row.factor`	a factor from the metadata to show along the rows of the heatmap (see Details below).
`dendro`	a character vector specifying on which axes (if any) a dendrogram should be plotted. Must be one of "`none`", "`both`", "`column`", or "`row`".
`top`	the number of groups to select, starting with the most abundant. If `NULL`, all are selected.

count	logical. Should the actual count of each OTU be shown, or should the relative abundances be shown?
drop.unclassified	logical. Should OTUs labelled "unclassified" or missing classification at the given taxonomic rank be excluded?
file	the file path where the image should be created (see ?RAM.plotting).
ext	the file type to be used; one of "pdf", "png", "tiff", "bmp", "jpg", or "svg".
height	the height of the image to be created (in inches).
width	the width of the image to be created (in inches).
leg.x	how far the legend should be inset, on the x axis.
leg.y	how far the legend should be inset, on the y axis.

Details

row.factor should be a named character vector specifying the name of the row to be used from meta (see RAM.factors). It should also be a factor; if it is not, a warning is raised and it is coerced to a factor (see factor). A warning is also raised when a factor has more than 8 levels, as that is the most colours the current palettes support. The factor must also correspond to the OTU table; i.e. they should have the same samples. If not, an error is raised.

Note

This function creates the heatmap using the heatmap.2 function from the gplots package. That function calls layout to set up the plotting environment, which currently prevents plotting two data sets side by side, or to automatically place the (metadata) legend. Unfortunately, this means that the leg.x and leg.y parameters must be used to adjust the legend by trial and error. It is possible to move the legend outside of the plotting area; if no legend appears, try with small leg.x and leg.y values.

Author(s)

Wen Chen and Joshua Simpson.

See Also

factor, heatmap.2, RAM.factors

Examples

```
data(ITS1, meta)

# plot the abundance of all observed classes for each sample, displaying it to
# the screen and adding a dendrogram on the column and the Collector on the row
group.heatmap.simple(ITS1, meta, rank="c", dendro="column",
     row.factor=c(City="City"))

## Not run:
# plot the genus for all OTUs, add a dendrogram to the row and column, and save
```

```
# the plot in path.tiff
group.heatmap.simple(ITS1, meta, rank="genus", dendro="both",
     file="my/file/path")
## End(Not run)
```

`group.indicators` *Plot Indicator Taxon Groups for Metadata Trends*

Description

This function conumes one or two OTU tables, along with a metadata facotr, and creates a barplot showing the relative abundance of all groups which are statistical indicators of that factor.

Usage

```
group.indicators(otu1, otu2 = NULL, meta, factor, rank,
                 thresholds = c(A = 0.85, B = 0.8, stat = 0.8, p.value = 0.05),
                 labels = c("ITS1", "ITS2"),
                 file = NULL, ext = NULL, height = 12, width = 12)
```

Arguments

`otu1`	the first OTU table to be used.
`otu2`	the second OTU table to be used.
`meta`	the metadata table to be used.
`factor`	a named character vector (of length 1) giving the name of the column in `meta` to be used when performing the analysis (see `RAM.factors`).
`rank`	the taxonomic rank to use (see ?RAM.rank.formatting for formatting details).
`thresholds`	a character vector of length 4 specifying the thresholds for the A, B, stat, and p values (see Details).
`labels`	a character vector giving the labels for the OTU tables.
`file`	the file path where the image should be created (see ?RAM.plotting).
`ext`	the file type to be used; one of `"pdf"`, `"png"`, `"tiff"`, `"bmp"`, `"jpg"`, or `"svg"`.
`height`	the height of the image to be created (in inches).
`width`	the width of the image to be created (in inches).

Details

This function uses `indicspecies::multipatt` to determine the indicators. After this analysis is performed, there will likely be some species determined to be 'significant,' but to varying degrees. To control how many groups are selected, you can adjust the `thresholds` argument. It consists of four components (quotations taken from `vignette("indicspeciesTutorial")`, see References):

A the *specificity* of the indicator; "the probability that the surveyed site belongs to the target site group given the fact that the species has been found."

B the *fidelity* of the indicator; "the probability of finding the species in sites belonging to the site group."

stat the association strength for the combinations.

p.value "the degree of statistical significance of the association."

Only the species with `A`, `B`, and `stat` values above, and `p.value` below those given in `thresholds` will be kept.

Author(s)

Wen Chen and Joshua Simpson.

References

De Caceres, M., Legendre, P. (2009). Associations between species and groups of sites: indices and statistical inference. Ecology, URL http://sites.google.com/site/miqueldecaceres/

See Also

`multipatt`

Examples

```
data(ITS1, ITS2, meta)

group.indicators(ITS1, ITS2, meta, factor = c(Province="Province"), rank="g")
```

`group.spatial` *Plot Spatial Collection Trends for Taxon Groups*

Description

This function consumes an OTU table and its associated metadata table, and uses that information to produce a choropleth (essentially a heatmap, but superimposed imposed on an actual, cartographic map) to show how many counts of each taxon group were detected in each Canadian province/territory.

Usage

```
group.spatial(data, meta, date.col, province.col, rank, group, breaks = "year",
              file = NULL, ext = NULL, height = 8, width = 10)
```

Arguments

data	the OTU table to be used.
meta	the metadata table to be used.
date.col	a character vector specifying which column in metadata contains the date information (see RAM.dates).
province.col	a character vector specifying which column in metadata contains the province information (see Details).
rank	a character vector specifying the rank of the desired taxon groups. Note that all groups should come the same rank (see RAM.rank.formatting).
group	a character vector giving the names of the groups to be plotted.
breaks	how many time segments should be plotted; see Details.
file	the file path where the image should be created (see ?RAM.plotting).
ext	the file type to be used; one of "pdf", "png", "tiff", "bmp", "jpg", or "svg".
height	the height of the image to be created (in inches).
width	the width of the image to be created (in inches).

Details

This function currently only supports Canadian data. The entries in meta$province.col should be specified as provinces, using the standard postal abbreviations (e.g. "Ontario" would be "ON").

The breaks argument is slightly buggy at the moment, possibly due to how R tries to split Date objects. breaks can be either an integer, in which case it will attempt to create that many levels (i.e. setting breaks=3 should split the data into three date 'blocks'.) breaks can also be a character vectors, such as "quarter" or "year" which attempts to split the date information accordingly. See cut.Date for more details and a complete specification of what is allowed for breaks.

Author(s)

Wen Chen and Joshua Simpson.

References

The file used to create the map of Canada is from GeoBase and is licensed under the Open Government License - Canada.

Examples

```
data(ITS1, meta)

## Not run:
group.spatial(ITS1, meta, date.col="Harvestdate", province.col="Province",
              rank="p", group=c("Ascomycota", "Basidiomycota"), breaks=2)
## End(Not run)
```

`group.temporal` *Plot Temporal Trends for Metadata and Taxon Groups*

Description

This function consumes an OTU table and its associated metadata, and creates a plot showing how the collections of taxonomic groups, as well as metadata factors, evolve over time.

Usage

```
group.temporal(data, meta, date.col, factors, rank, group,
               file = NULL, ext = NULL, height = 8, width = 12)
```

Arguments

`data`	the OTU table to be used.
`meta`	the metadata table to be used.
`date.col`	a character vector specifying which column of the metadata has date information (see `RAM.dates`).
`factors`	a named character vector specifying the names of the metadata columns to be plotted with the taxon group data (see `RAM.factors`). NOTE: these factors must be *numeric* variables.
`rank`	a character vector specifying the rank of the desired taxon groups. Note that all groups should come the same rank (see `RAM.rank.formatting`).
`group`	a character vector giving the names of the groups to be plotted.
`file`	the file path where the image should be created (see ?RAM.plotting).
`ext`	the file type to be used; one of `"pdf"`, `"png"`, `"tiff"`, `"bmp"`, `"jpg"`, or `"svg"`.
`height`	the height of the image to be created (in inches).
`width`	the width of the image to be created (in inches).

Details

The image created will contain several plots. It will always contain a large panel showing the counts collected for the specified taxon groups over time, and above that panel (on a common x-axis) will be a line graph for each metadata factor specified.

Note

If your data has collections being taken roughly annually, you may have a large amount of "empty space" in the middle of your plot. Consider subsetting the data by year, and plotting each year individually using this function.

Author(s)

Wen Chen and Joshua Simpson

Examples

```
data(ITS1, meta)

group.temporal(ITS1, meta, date.col="Harvestdate", factors=c(Ergosterol="Ergosterol_ppm"),
               rank="p", group=c("Ascomycota", "Basidiomycota"))
```

ITS1/ITS2 *Sample ITS1 and ITS2 Data*

Description

Sample ITS1 and ITS2 OTU tables.

Usage

```
data(ITS1)
data(ITS2)
```

Format

A data frame with 4704 observations on the following 17 variables.

`P1001.1M1, P1001.1M2, P1001.1M3, P1001.1M4, P1001.1M5, P1001.1M6, P1001.1M7, P1001.1M8, P1001.1M9,`
Collection samples.

`taxonomy` the taxonomic classification of the OTU.

Source

Wen Chen, AAFC-AAC

Examples

```
data(ITS1, ITS2)

str(ITS1)
str(ITS2)
```

location.formatting *Location Formatting*

Description

Some functions in RAM expect to find a column with provincial/territorial data in the metadata. This data should use the standard Canadian provincial/territorial abbreviations:

- Alberta - AB
- British Columbia - BC
- Manitoba - MB
- New Brunswick - NB
- Newfoundland and Labrador - NL
- Novia Scotia - NS
- Northwest Territories - NT
- Nunavut - NU
- Ontario - ON
- Prince Edward Island - PE
- Quebec - QC
- Saskatchewan - SK
- Yukon - YT

Support for other locations is not available at this time.

Author(s)

Wen Chen and Joshua Simpson.

meta *Sample Metadata for ITS1/ITS2*

Description

This data frame provides sample metadata for the ITS1/ITS2 data included in this package.

Usage

```
data(meta)
```

Format

A data frame with 16 observations on the following 10 variables.

`Sample` a factor with levels `Sample1` `Sample2` `Sample3` `Sample4` `Sample5` `Sample6` `Sample7` `Sample8`

`City` a factor with levels `City1` `City2`

`Crop` a factor with levels `Crop1` `Crop2`

`Plots` a factor with levels `1` `2`

`Harvestmethod` a factor with levels `Method1` `Method2`

`Harvestdate` a Date

`Ergosterol_ppm` a numeric vector

`Province` a character vector

`Latitude` a numeric vector

`Longitude` a numeric vector

Source

Wen Chen and Joshua Simpson.

Examples

```
data(meta)

str(meta)
```

---

pcoa.plot *Create a PCoA plot for an OTU Table*

Description

This function consumes an OTU table, metadata factors, and graphing options, then produces a plot showing the PCoA analysis of the OTU table.

Usage

```
pcoa.plot(data, meta, factors, rank, stand.method = NULL,
          dist.method = "morisita", sample.labels = TRUE, top = 20,
          ellipse = FALSE, file = NULL, ext = NULL, height = 8,
          width = 10, ggplot2 = TRUE, bw = FALSE)
```

Arguments

data	the OTU table to be used.
meta	the metadata table to be used.
factors	a named character vector of length 1 or 2 specifying metadata factors for the samples in the OTU table (see Details).
rank	the rank to select the taxon groups at.
stand.method	a method used to standardize the OTU table. One of "total", "max", "freq", "normalize", "range", "standardize", "pa", "chi.square", "hellinger" or "log" (see ?decostand).
dist.method	the dissimilarity index to be used; one of "manhattan", "euclidean", "canberra", "bray", "kulczynski", "jaccard", "gower", "altGower","morisita", "horn", "mountford", "raup", "binomial","chao", or "cao" (see vegdist).
sample.labels	logical. Should the labels for the samples be displayed?
top	how many taxon groups should be displayed, starting from the most abundant.
ellipse	which of the metadata factors (if any) should have ellipses plotted around them. Must be one of 1, 2, or FALSE (see Details).
file	the file path where the image should be created (see ?RAM.plotting).
ext	the file type to be used; one of "pdf", "png", "tiff", "bmp", "jpg", or "svg".
height	the height of the image to be created (in inches).
width	the width of the image to be created (in inches).
bw	logical. Should the image be created in black and white?
ggplot2	logical. Should the ggplot2 package be used to produce the plot, or should the base graphics be used? (see ?RAM.plotting).

Details

factors should be a named character vector specifying the names of the columns to be used from meta (see RAM.factors). Those columns should be factors; if they are not, a warning is raised and they are coerced to factors (see factor). A warning is also raised when a factor has more than 9 levels, as that is the most colours the current palettes support.

The values on the axes denote what fraction of the sum of all eigenvalues (i.e. from all axes) is explained by that (single) axis.

When ellipse = FALSE, no ellipses will be plotted. When ellipse is a number, that 'number' metadata factor will have ellipses plotted. For example, if factors = c(Crop="Crop", City="City") and ellipse = 1, ellipses will be plotted for the different crops, but NOT the cities. Setting factors = c(City="City") and ellipse = 2 is invalid, since there is no second metadata factor given. Ellipses can only be plotted for one factor currently. Furthermore, there need to be at least 3 samples for every level in every item in factors, otherwise ellipses cannot be plotted.

Value

When ggplot2 = TRUE, a ggplot object is returned; otherwise nothing is returned (but the plot is shown on screen).

Note

The labels for the sample points are placed above, below, or next to the point itself at random. If labels are outside of the plotting area, or overlapping with each other, run your command again (without changing any arguments!) and the labels should move to new positions. Repeat until they are placed appropriately. This is done to ensure even tightly-grouped samples, or samples near the edge of the plot, have their labels shown. If the labels are too distracting, remember that they can be turned off by setting `sample.labels = FALSE`.

Author(s)

Wen Chen and Joshua Simpson.

See Also

vegdist

Examples

```
data(ITS1, meta)

# The argument for factors is a vector of length two; the first item is
# Crop, which is a column from meta, and the second item is City, another
# column from meta.
pcoa.plot(ITS1, meta, rank="c", factors=c(Crop="Crop", City="City"))

# If you want to customize legend labels and plot the top 20 taxon groups at genus:
pcoa.plot(ITS1, meta, rank="g",
          factors=c(Place="City", Harvest_Method="Harvestmethod"))

## Not run:
# In black & white, using base graphics:
pcoa.plot(ITS1, meta, rank="c", factors=c(Plot="Plots"), ggplot=F, bw=T)

# Focus on the samples: hide all groups and plot ellipses for Crop:
pcoa.plot(ITS1, meta, rank="g", factors=c(Crop="Crop", City="City"),
          ellipse=1, sample.labels=FALSE, top=0)

# Standardize the data before calculating distances:
pcoa.plot(ITS1, meta, rank="g", factors=c(City="City"),  stand.method="chi.square",
          dist.method="euclidean")
## End(Not run)
```

---

percent.classified *Calculate Percent of OTUs Classified at a Given Taxonomic Rank*

Description

This function consumes an OTU table, and a rank, then returns what percent of OTUs in the given table are classified at that taxonomic rank.

Usage

```
percent.classified(data, rank)
```

Arguments

data	the OTU table to be processed.
rank	the taxonomic rank you are interested in (see ?RAM.rank.formatting for formatting details).

Value

A numeric vector of length one, containing the result.

Author(s)

Wen Chen and Joshua Simpson.

Examples

```
data(ITS1)

# find what percent of OTUs classified at genus level
percent.classified(ITS1, "g")
```

RAM.dates *Date Formatting for RAM*

Description

This help page details the expected format for dates in RAM.

Details

For all functions expecting some type of date data, you will need to specify which column of the metadata table contains that information. The date information will likely be encoded as a character vector from read.meta, so these functions will try to coerce it to a Date object (see Date and as.Date), with a warning. These functions are expecting the date information to be in YYYY-MM-DD format.

RAM.factors *Factor Formatting for RAM*

Description

This help page details how to pass metadata arguments in RAM.

Details

Many functions will expect arguments such as `meta` and `factors` (possibly `row.factor` or `col.factor`). These functions are expecting the full metadata table for `meta` (which you probably read into R using `read.meta`). The other argument, `factor`, should be a *named* character vector. The values of this vector should be the columns to be used from `meta`, while the names of the vector should be the labels you wish to have displayed in the plots. There are several ways to name a character vector:

> my.vec <- c(This = "is", a = "named", character = "vector")

> names(my.vec) [1] "This" "a" "character"

> cat(my.vec) is named vector

Notice that `myvec` has *names* "`This`", "`a`", "`character`", but has *values* "`is`", "`named`", "`vector`". It is the names that will be used to label graphs in `RAM`, but the values that will be used to extract the actual data. This is useful if you have more complicated names; say we want the data from the column named "Precip_14d_before_harvest", but we want a nicer label for the plot. We can do the following:

> my.vec <- "Precip_14d_before_harvest"

> names(my.vec) <- "Precipitation (14 d. prior to Harvest, mm)"

Now we will be able to plot the value of the "`Precip_14d_before_harvest`" column, but we will have the (much nicer!) label "Precipitation (14 d. prior to Harvest, mm)" appear in our plots.

RAM.plotting *Creating Plots with RAM*

Description

This help page details the standards for RAM plotting functions.

Overview

The RAM package contains many functions to produce plots and visualizations for metagenomic data. Currently, the plotting functions are grouped into 'casual' and 'publication' categories. The 'casual' plotting functions only accept a `file` argument and produce a `.tiff` file automatically. They are meant to quickly highlight certain aspects of the data, but are not meant to be published. The 'publication' quality plots accept many more graphing parameters, and should be of suitable quality for future publication. All 'publication' plots should accept the following parameters, in addition to those required to produce the plot:

- "file" the file name for the plot.
- "ext" the file type for the plot (see below).
- "height" the height of the plot, in inches.
- "width" the width of the plot, in inches.

Additionally, the following parameters are accepted by some functions:

- "bw" should the plot be in black and white?

For 'casual' plots, if file is not provided, the plot is displayed to the default graphics device (usually a new window), otherwise a .tiff file is created.

For 'publication' plots, if neither file nor ext are provided, the plot is displayed to the default graphics device (usually a new window). If both file and ext are provided, a file of type ext is created at file. If only one of file or ext is given, an error is raised.

In either case, the file extension will automatically be appended to the end of file, if file does not already end in the appropriate extension. For example, if file = ~/my/path.tiff and ext = png, the file will be called ~/my/path.tiff.png; but if file = ~/my/path.png, the file will just be called ~/my/path.png. Finally, if file = ~/my/path, the file will be called ~/my/path.png.

ggplot2

Furthermore, some of the 'publication' quality plots allow for a ggplot2 argument. If ggplot2 is TRUE, then the plot will be produced using the ggplot2 package, and a ggplot object will be returned. This allows for additional, personal customization of the plot for those who have used the package before. If ggplot2 is FALSE, then the plot will be created using the base plotting functions.

File Types

For 'publication' quality plots, the following file types are supported (use any of the following values for the ext argument):

"pdf", "png", "tiff", "svg", "bmp", "jpeg".

Note

If file is given without a directrory (e.g. file = my_fancy_file), then that file will be created in the current working directory (see ?getwd and ?setwd for more information).

The current default resolution is 1000 dpi for all plots.

See Also

ggplot

Author(s)

Wen Chen and Joshua Simpson.

RAM.rank.formatting *Taxonomic Rank Formatting*

Description

In all RAM functions requiring the user to input a taxonomic rank, three different formats for this taxon are accepted. All of these formats are simple character vectors (strings), and are provided for readability and convenience. The user only needs to specify any single element from any of the formats below. The formats are as follows:

Format 1: "kingdom", "phylum", "class", "order", "family", "genus", "species"

Format 2: "k", "p", "c", "o", "f", "g", "s"

Format 3: "k__", "p__", "c__", "o__", "f__", "g__", "s__"

Author(s)

Wen Chen and Joshua Simpson.

See Also

get.rank, tax.abund

read.meta *Open Metadata Table*

Description

Opens the given file and return a data frame representing the metadata.

Usage

```
read.meta(file, sep=",")
```

Arguments

file	a character vector specifying the file path to your file.
sep	the character used to separate cells in the file.

Value

Returns a data frame with the information from the file. The first row and column are used for the names of the other rows and columns.

Author(s)

Wen Chen and Joshua Simpson

See Also

getwd, setwd, read.table

Examples

```
## Not run:
my.meta <- read.meta("path/to/meta")
## End(Not run)
```

---

read.OTU *Open OTU Table*

Description

Opens the given file and returns a data frame representing the OTU table.

Usage

```
read.OTU(file, sep=",")
```

Arguments

file	a character vector specifying the file path to your file.
sep	the character used to separate cells in the file.

Value

Returns a data frame with the information from the file. The first row and column are used for the names of the other rows and columns.

Note

The OTU table should only contain classifications for the seven major taxonomic ranks, additional ranks will break some functions in the package. To remove those other classifications, try running

```
sed -i.backup -e 's/s[a-z]__[^;]*; //g' -e 's/t__[^;]*; //g' $FILE
```

where $FILE is your OTU table. The letter t can be replaced in the second expression for any other letter which appears as a prefix. For example, adding -e 's/u__[^;]*; //g' before $FILE would remove any entries formatted like u__test_classification; .

Additionally, if your OTU table starts with the entry #OTU ID, that cell needs to be removed before the table can be read with read.OTU.

Author(s)

Wen Chen and Joshua Simpson.

See Also

getwd, setwd, read.table

Examples

```
## Not run:
my.OTU <- read.OTU("path/to/data")
## End(Not run)
```

---

sample.locations *Plot the Geographic Location of Samples*

Description

This function consumes an OTU table, along with its associated metadata, and plots all the samples from that data as points on a map. The size of a point indicates the number of counts collected from that sample, while the colour of the point (optionally) shows a metadata factor for that sample.

Usage

```
sample.locations(otu1, otu2=NULL, meta, factor = NULL, zoom = 5,
                 source = "google", labels = c("ITS1", "ITS2"),
                 lat.col = "Latitude", long.col = "Longitude",
                 file = NULL, ext = NULL, height = 10, width = 12)
```

Arguments

otu1	the OTU table to be used.
otu2	the (optional) second OTU table to be used.
meta	the metadata table to be used.
factor	(optional) a named character vector of length one specifying a column from the metadata table to be used to colour the points.
zoom	a positive integer in the range 3-21 (if source == "google") or 3-18 (if source == "osm") specifying the zoom for the map (low number means zoomed out).
source	the source to be used for the map; either "google" or "osm".
labels	a character vector giving one label per OTU.
lat.col	the name of the column in meta containing the latitude information.
long.col	the name of the column in meta containing the longitude information.
file	the file path where the image should be created (see ?RAM.plotting).
ext	the file type to be used; one of "pdf", "png", "tiff", "bmp", "jpg", or "svg".
height	the height of the image to be created (in inches).
width	the width of the image to be created (in inches).

Details

Please note that this function is getting map information using either the Google Maps API or the OpenStreetMap API, and your usage is subject to the terms of those APIs.

Note

If you are getting a 403/503 error, that likely means that the current map provider is currently unavailable. This can be for a variety of reasons: if `source == "google"`, you have likely maxed out your API call limit (this can be due to multiple users sharing an IP address; contact your system administrator for further details). If `source == "osm"`, the server is likely under heavy load and unable to process your request. You can check the server load online. In either case, the issue will likely resolve itself. Try calling the function again in a few hours.

If you get a warning message of the form "`Removed X rows containing missing values (geom_point).`", this means that the current zoom level is too high to display some or all of the points. Try using a lower value for `zoom`.

Author(s)

Wen Chen and Joshua Simpson.

See Also

`RAM.factors`

Examples

```
data(ITS1, meta)

## Not run:
sample.locations(ITS1, meta, factor=c(Crop="Crop"))
## End(Not run)
```

---

shared.OTU *Show Summary of Shared OTUs*

Description

This function consumes an OTU table and returns a list summarizing information about the presence of the OTUs in that table.

Usage

```
shared.OTU(data)
```

Arguments

`data`	the OTU table to be analyzed.

Value

shared.OTU returns a list containing the information calculated. The names associated with the list describe what that number represents; i.e. "#_of_OTUs_in_all_samples" shows how many OTUs in the given table were found to be present in all samples. The last item in the list is a character vector, containing the OTU number and taxonomic information of each OTU which was present in all samples. All entries in that column are of the form "OTU-taxonomic_classification".

Note

The OTUs are determined to be absent/present using the "pa" method from the function decostand.

Author(s)

Wen Chen and Joshua Simpson.

See Also

decostand

Examples

```
data(ITS1)

shared.OTU(ITS1)
```

---

tax.abund *Aggregate OTU Data Based on Taxonomy*

Description

This function consumes OTU table(s) and (optionally) a taxonomic rank, then extracts the classification of each OTU at the given taxonomic rank, groups by classification at the given rank, removes all groups with only 0 counts, optionally removes all unclassified groups, sorts groups based on abundance, and then returns the transpose.

Usage

```
tax.abund(otu1, otu2=NULL, rank=NULL, drop.unclassified=FALSE,
          top=NULL, count=TRUE, mode="number")
```

Arguments

otu1	the first OTU table to be used.
otu2	the second OTU table to be used.
rank	a character vector representing a rank. Must be in one of three specific formats (see ?RAM.rank.formatting for help). If omitted, the function will repeat for all seven major taxonomic ranks.

`drop.unclassified`
: logical. Should OTUs labelled "unclassified" or missing classification at the given taxonomic rank be excluded?

`top`
: the number of groups to select, starting with the most abundant. If `NULL`, all are selected.

`count`
: logical. Should the actual count of each OTU be shown, or should the relative abundances be shown?

`mode`
: a character vector, one of "`percent`" or "`number`". If `number`, then `top` many groups will be selected. If `percent`, then all groups with relative abundance in at least one sample above `top` will be selected.

Value

The value returned by this function may become nested lists, so for convenience, any nested lists have been given descriptive items names to make accessing its elements simple (see Examples).

- If `otu2` is given:
  - If `rank` is given: a list containing two data frames (otu1 and otu2 aggregated at the given rank).
  - If `rank` is not given: a list containing two lists. The first sublist represents `otu1`, the second `otu2`. The sublists contain seven data frames, the aggregation of the data at each taxonomic rank (see Examples).
- If `otu2` is not given:
  - If `rank` is given: a single data frame (otu1 aggregated at the given rank).
  - If `rank` is not given: a list containing seven data frames (otu1 aggregated at each taxonomic rank).

Author(s)

Wen Chen and Joshua Simpson.

See Also

`RAM.rank.formatting`

Examples

```
data(ITS1, ITS2)
# aggregate based on phylum
ITS1.p <- tax.abund(ITS1, rank="p")

# aggregate based on all ranks; ignoring all unclassified OTUs
ITS1.taxa <- tax.abund(ITS1, drop.unclassified=FALSE)

# aggregate for one rank for both ITS1 and ITS2
lst <- tax.abund(ITS1, ITS2, rank="class")

# aggregate for all ranks for both ITS1 and ITS2
lst.all <- tax.abund(ITS1, ITS2)
```

```
stopifnot(identical(lst.all$otu1$phylum, ITS1.p))

# get the counts for all genera with relative abundance > 25%
tax.abund(ITS1, rank="g", top=25, mode="percent", count=TRUE)
```

`tax.fill` *Fill Missing Taxonomic Information*

Description

This function consumes an OTU table and returns a new OTU table where the taxonomic classifications which are unidentified, unclassified, incertae sedis, or simply missing, are replaced with a more descriptive entry.

Usage

```
tax.fill(data, downstream = TRUE)
```

Arguments

`data`	the OTU table to be used.
`downstream`	logical. Should the replacement occur downstream, or upstream? (see Details)

Details

If `downstream == TRUE`, the function will start at the kingdom level and work its way down. Whenever an invalid group is encountered (i.e. one of "unclassified", "unidentified", "incertae_sedis", or simply missing, ignoring capitalization), the last known 'good' group will be substituted in the form "`p__belongs_to_k_Fungi`." If `downstream == FALSE`, the function will begin at the species level and work up.

This example should help clarify: given the taxonomy "k__Fungi; p__unidentified; c__Tremellomycetes", the new taxonomy is as follows (recall that an OTU table is required as input, and will be returned as output; this example simply shows the effect on the taxonomy):

- Downstream (Kingdom -> Species): "k__Fungi; p__belongs_to_k_Fungi; c__Tremellomycetes; o__belongs_to_c_Tremellomycetes; f__belongs_to_c_Tremellomycetes; g__belongs_to_c_Tremellomycetes; s__belongs_to_c_Tremellomycetes"
- Upstream (Species -> Kingdom): "k__Fungi; p__belongs_to_c_Tremellomycetes; c__Tremellomycetes; o__belongs_to_no_taxonomy; f__belongs_to_no_taxonomy; g__belongs_to_no_taxonomy; s__belongs_to_no_taxonomy"

Usually, `downstream = TRUE` will provide a more useful output, however if the species is often known for your data, but other ranks are unknown, `downstream = FALSE` will provide a more descriptive taxonomy.

Value

Returns an OTU table as a data frame with the exact same numerical counts as `data`, but an updated `taxonomy` column.

Author(s)

Wen Chen and Joshua Simpson.

See Also

`RAM.rank.formatting`

Examples

```
data(ITS1)

tax.fill(ITS1)
```

`tax.split` *Split OTU Tables By Taxonomic Rank*

Description

This function consumes an OTU table and splits its taxonomy columns into the seven major taxonomic ranks. It returns a data frame preserving all numerical data, but changing the 'taxonomy' column to the name of the appropriate rank, and preserving only the classifications at that rank.

Usage

```
tax.split(otu1, otu2 = NULL, rank = NULL)
```

Arguments

`otu1`	the first OTU table to be used.
`otu2`	the second OTU table to be used.
`rank`	the (optional) rank to split at and return (see ?RAM.rank.formatting for formatting details).

Value

The value returned by this function may become nested lists, so for convenience, any nested lists have been given descriptive items names to make accessing its elements simple (see Examples).

- If `otu2` is given:
  - If `rank` is given: a list containing two data frames (otu1 and otu2 split at the given rank).

  - If rank is not given: a list containing two lists. The first sublist represents otu1, the second otu2. The sublists contain seven data frames, which are the data split at each taxonomic rank (see Examples).

- If otu2 is not given:
  - If rank is given: a single data frame (otu1 split at the given rank).
  - If rank is not given: a list containing seven data frames (otu1 split at each taxonomic rank).

Note

This function may seem similar to get.rank, but they are distinct. get.rank only returns the entries classified at that taxonomic rank, and so some OTUs might be omitted in the returned data frame. With tax.split, it is guaranteed that all OTUs will be preserved in the returned data table (although they may be missing taxonomic classification at that rank).

If no OTUs are classified at the given rank, the taxonomy column for that rank will be filled with empty strings.

Author(s)

Wen Chen and Joshua Simpson.

See Also

get.rank

Examples

```
data(ITS1, ITS2)

# split only ITS1 data at a single rank
ITS1.tax.p <- tax.split(ITS1, rank="phylum")

# split only ITS1 data at all ranks
ITS1.tax.all <- tax.split(ITS1)

# split ITS1 and IST2 data at a given rank
lst <- tax.split(ITS1, ITS2, rank="c")

# split ITS1 and ITS2 at every rank
lst.all <- tax.split(ITS1, ITS2)

stopifnot(identical(lst.all$otu1$phylum, ITS1.tax.p))
```

`top.groups.plot` *Plot the Top Taxon Groups*

Description

These functions consume two OTU tables, along with (optionally) a file name and a parameter `top`. They create a box plot of the `top` number of OTUs (for `plot.top.number`), or all OTUs with relative abundance above `top` percent (for `plot.top.percent`) at the taxonomic ranks phylum, class, order, family, and genus.

Usage

```
group.top.number(otu1, otu2=NULL, top=10, drop.unclassified=FALSE,
                 labels=c("ITS1", "ITS2"),
                 file=NULL, ext=NULL, height=8, width=16,
                 bw=FALSE, ggplot2=TRUE)

group.top.percent(otu1, otu2=NULL, top=10, drop.unclassified=FALSE,
                  labels=c("ITS1", "ITS2"),
                  file=NULL, ext=NULL, height=8, width=16,
                  bw=FALSE, ggplot2=TRUE)
```

Arguments

`otu1`	the first OTU table to be used.
`otu2`	the second OTU table to be used.
`top`	the number of OTUs to select (for `top.number`), or the minimum relative abundance threshold to use for selecting OTUs (for `top.percent`).
`drop.unclassified`	logical. Should OTUs labelled "unclassified" or missing classification at the given taxonomic rank be excluded?
`labels`	a character vector giving one label per OTU.
`file`	the file path where the image should be created (see ?RAM.plotting).
`ext`	the file type to be used; one of `"pdf"`, `"png"`, `"tiff"`, `"bmp"`, `"jpg"`, or `"svg"`.
`height`	the height of the image to be created (in inches).
`width`	the width of the image to be created (in inches).
`bw`	logical. Should the image be created in black and white?
`ggplot2`	logical. Should the `ggplot2` package be used to produce the plot, or should the base graphics be used? (see ?RAM.plotting).

Note

Please be aware that the 'whiskers' in the plot may differ depending on the setting of `ggplot2`. Please see `geom_boxplot` and `boxplot`/`boxplot.stats` for more information on how the whiskers are calculated.

Author(s)

Wen Chen and Joshua Simpson.

See Also

RAM.plotting

Examples

```
## Not run:
data(ITS1, ITS2)

# plots the top 10 OTUs (by default) at five ranks
group.top.number(ITS1, ITS2)

# plots all OTUs w/ relative abundance > 10% (as specified) at five ranks
# and saves the result as a .tiff file (only using ITS1 data)
group.top.percent(ITS1, top=10, file="my/file/path", ext="tiff")
## End(Not run)
```

---

transpose.OTU *Take the Transpose of an OTU Table*

Description

Returns the transpose of the given OTU table, excluding the last column (which should contain taxonomic information).

Usage

```
transpose.OTU(data)
```

Arguments

data	The OTU table to be transposed.

Value

Returns a data frame with rows equal to the columns of the original OTU, and columns equal to the rows of the original OTU. (Excluding the taxonomy column).

Author(s)

Wen Chen and Joshua Simpson.

Examples

```
data(ITS1)

ITS1.t <- transpose.OTU(ITS1)
```

valid.OTU *Validate an OTU Table*

Description

This function consumes one or two OTU tables and checks if they are formatted properly and contain valid data.

Usage

```
valid.OTU(otu1, otu2 = NULL)
```

Arguments

otu1	the first OTU table to check.
otu2	the second OTU table to check.

Value

If the table is not valid, an error will be raised with a description explaining the problem. If the table is valid, NULL will be returned invisibly.

Author(s)

Dr. Chen Wen and Joshua Simpson.

Examples

```
data(ITS1, ITS2)

valid.OTU(ITS1)
valid.OTU(ITS1, ITS2)
```

write.OTU *Write OTU-Style Table to .csv File*

Description

Creates a .csv-formatted file with the data from the specified OTU table data. The file will be created in your current working directory (see ?getwd and ?setwd), unless specified otherwise by file. Note that if the file field does not end in .csv, ".csv" will be appended to the end of file.

Usage

```
write.OTU(data, file)
```

Arguments

data	The OTU table containing the data to be used.
file	The name of the .csv file to be created.

Author(s)

Wen Chen and Joshua Simpson.

See Also

write.csv, getwd, setwd

Examples

```
data(ITS1)
## Not run:
write.OTU(ITS1, "my_file_name.csv")
## End(Not run)
```

Index